

# 모바일 게임프로그래밍

-강의자료 요약-

과목명 : 모바일 게임프로그래밍

담당 교수님 : 배재환 교수님

학과 : 게임공학과

학번 : 19110171

이름 : 윤 건

제출일 : 2023.04.14.

## 목차

1. 모바일 게임의 개요
2. 모바일 게임 설계사례 분석
3. Unity 기반의 모바일 게임 프로그래밍
4. Unity 사용법
5. 게임 시스템 아키텍처 소개
6. 모바일 게임 개발
7. 게임의 분류
8. 게임 플랫폼
9. Unity 조명 시스템
10. 게임 장면 시스템

# 1. 모바일 게임의 개요

모바일 게임 : 넓은 의미로는 모바일 기기에서 이용하는 게임이라 할 수 있고, 좁은 의미로는 휴대폰에 내장되어있는 게임이나 이용자가 휴대폰의 무선 인터넷에 접속하여 다운을 받아 이용하는 게임이라 할 수 있다.

모바일 게임은 특성상 시간, 공간적 제약을 받지 않고 게임을 즐길 수 있으며 타 플랫폼에 비해 휴대성, 간편성이 뛰어나다고 할 수 있다. 조작법 또한 간단하여 매니아 그룹 없이 보편적으로 쉽게 즐길 수 있는 장점이 있다.

초기에는 하드웨어 성능이나 소프트웨어의 한계로 다채롭지 못하고 조작이 간단한 퍼즐 게임들이 주류였으나, 최근에는 하드웨어 성능 및 LTE가 널리 보급되며 다양한 장르의 게임들이 구현되고 있다.

## - 모바일 게임의 장르별 특성

ARPPU (Average Revenue Per Playing User)

↳ 과금 유저 당 결제금액

### 1) RPG

포함 장르 : 액션 RPG, MMORPG, 턴제 RPG

개발 비용 : 높음

플레이 시간 : 높음

ARPPU : 높음

### 2) Casual

포함 장르 : 보드, 퍼즐

개발 비용 : 낮음

플레이 시간 : 낮음

ARPPU : 낮음

### 3) Action, Sport

포함 장르 : 플랫폼, 슈팅, 대전, 스포츠

개발 비용 : 높음

플레이 시간 : 중간

ARPPU : 중간

#### 4) 전략/시뮬레이션

포함 장르 : 실시간/턴 베이스, TCG

개발 비용 : 높음

플레이 시간 : 상대적 높음

ARPPU : 상대적 높음

#### 5) 소셜 카지노

포함 장르 : 카지노 및 빙고

개발 비용 : 낮음

플레이 시간 : 중산

ARPPU : 중간

### -수익배분 구조

수익이 발생 했을 때, 1차로 플레이 스토어나 앱스토어 등의 OS 플랫폼에서 총수익의 30%를 가져간다. 여기서 만약 라인이나 카카오톡 같은 플랫폼을 사용했다면 2차로 수익을 떼어간다. 그리고 만약 퍼블리셔도 있다면 3차로 또 떼어간다. 예를 들어 100억을 벌었을 때 플랫폼에 라인을 사용하고 퍼블리셔까지 다 사용했다면 17.5억 밖에 가져가지 못한다.

그렇다고 아무것도 안 쓰는 것이 좋은 것이 아니다. OS플랫폼 같은 경우 안 쓸 수가 없고, 플랫폼의 경우 접근성을 좋게 해주며, 퍼블리셔의 경우 마케팅이나 이름만으로 사람을 끌어올 수 있기 때문에 이점도 많다. 어떤 것을 쓰고 어떤 것을 안 써도 되는지 잘 조율해야 할 필요성이 있다.

## 2. 모바일 게임 설계사례 분석

### -ELSION (교수님이 제작에 참여했던 게임)

이 게임의 기획 당시 PPT를 통해 기획에 필요한 점을 볼 수 있다.

#### 1) 게임 개요

게임 개요에는 타이틀 화면, 게임명 등이 들어가며 지원할 플랫폼(스마트폰 OS 버전 등), 개발에 사용할 엔진, 장르, 개발적용 폰 등 전반적인 게임의 정보들이 포함된다.

#### 2) 기획 의도

게임을 기획하게 된 계기와 방향성 등이 포함된다.

#### 3) 유사 경쟁 게임

내가 만들 게임과 유사한 게임들을 예시를 들며 해당 게임들의 특징이나 장점들이 포함된다.

#### 4) 시놉시스

게임에 포함된 세계관, 스토리 진행 방향 등을 짧게 표현한다.

#### 5) 게임 특징

게임에 포함될 게임만의 특징을 표현한다. 특별한 콘텐츠, 기대 효과 등이 있고, 게임을 진행하면서 있는 시스템적인 부분들을 상세하게 표현한다.

#### 6) 게임 흐름도

게임 시작부터 엔딩까지의 순서도를 표현한다. 해당 순서마다 들어갈 메뉴, 시스템 등을 간단히 표현하고 순환 구조가 있을 경우도 나타낸다.

#### 7) 게임 조작법

게임 진행 중에 사용되는 UI의 간단한 설명과 게임 조작법을 나타낸다.

#### 8) 게임 인터페이스

게임의 인터페이스를 단계별로 화면을 예시로 들어 보여준다.

#### 9) 게임 그래픽

게임에 사용될 그래픽 요소들의 특징들을 예시를 들어 어떤 구조인지 보여준다.

#### 10) 게임 캐릭터

게임에 사용되는 캐릭터들의 특징이나 주요 유닛 등을 설명한다.

#### 11) 게임 아이템

게임의 아이템 구조를 보여준다. 아이템 등급, 형식, 획득 구조 등을 보여준다.

#### 12) 수익모델

게임을 운영하면서 수익이 나는 모델들을 보여준다. 어떤 부분에서 수익이 발생하는지, 예상 연수익이 얼마인지 보여준다.

#### 13) 예상 일정

게임 런칭 목표가 언제인지, 월별로 테스트 기간을 포함해서 상용화가 언제쯤 될지 보여준다.

### 3. Unity3D 기반의 모바일 게임 프로그래밍

#### - 게임 엔진

##### 1) 게임 엔진의 개요

- 게임과 같이 리얼타임으로 화면에 출력하는 소프트웨어를 제작하기 위한 요소로 구성된 소프트웨어를 말한다.

- 미들웨어

- 게임 제작에 들어가는 시간과 노력을 절약하기 위해 만든 각각의 기능 프로그램
- 이러한 미들웨어가 하나로 합쳐진 종합 프로그램이 게임 엔진의 시작

- 최신 범용 게임 엔진이 가지고 있는 일반적인 특징

- 물리 엔진
- 사운드 엔진
- 네트워크 엔진
- 스크립트 또는 맵 에디터

##### 2) 게임 엔진의 종류 (대표적인 게임 엔진)

- 유니티 엔진

- 덴마크에서 제작된 범용 게임 개발 엔진이다. 다양한 플랫폼을 지원한다.

- 언리얼 엔진

- 언리얼 게임을 제작하면서 외부에 공개한 엔진이다. 고품질 범용 엔진이다.

- 소스 엔진

- 밸드 코퍼레이션에서 제작한 3D 엔진이다. 주로 PC 게임 제작에 사용된다.

- 크라이 엔진

- 크라이텍에서 개발한 엔진이다. 야외 표현에 강점이 있다.

### 3) 게임 엔진의 활용 (게임 외 영역에서의 활용)

#### ● 자동차

- 리얼타임 쇼룸
- 사전 지원 교육

#### ● 건축

- 가상 모델 하우스

#### ● 방송

- 라이브 이벤트
- 가상 스튜디오

#### ● 영화 & TV

- 비용 절감
- 실시간 협업

#### - 유니티 엔진

다비드 헬가손, 요아킴 안테, 니콜라스 프란시스가 개발한 게임 엔진이며, 모바일 게임 시장의 부상으로 크게 성공했다.

#### - 유니티 엔진의 장점 및 특징

- \* 쉬운 통합 개발 환경
- \* 저렴한 가격
- \* 멀티플랫폼 지원
- \* 에셋 스토어

#### - 유니티의 기본 용어

##### ● 프로젝트

게임 전체를 의미

##### ● 씬

유저에게 보여주는 장면을 의미

##### ● 컴포넌트

에셋이 가지고 있는 특정한 기능 블록

##### ● 에셋

게임 개발을 위해 사용되는 모든 요소

## 4. Unity 사용법

### 1) Project 창

#### ● 프로젝트 창

- 게임 전체에서 사용하는 스크립트, 사운드, 텍스처, 모델링 등 모든 리소스가 들어있는 창이다.
- 이러한 데이터를 Asset이라고 하며, 게임에 사용되는 모든 데이터는 Assets 폴더 안에 존재해야 한다.

-

#### ● Asset 폴더 오른쪽클릭 후 [Show in Explorer] 선택하기

- 윈도우 탐색기로 볼 수 있음
- 탐색기에서 파일을 수정하는 것은 권장하지 않음
- 창의 위 부분의 + 버튼을 클릭하면 스크립트, 셰이더, 메터리얼, 애니메이션 컨트롤러 등의 에셋 생성 가능

#### ● Package 폴더

- 일종의 플러그인과 같이, 엔진 내부에서 사용되는 기능 모음
- 사용자가 접근하여 수정하거나 사용할 수 없는 영역
- 패키지 매니저를 통해 기능을 설치하거나 삭제
- One 칼럼 레이아웃과 Two 칼럼 레이아웃 스타일 선택 가능

### 2) Hierarchy 창

#### ● 현재 씬에 있는 모든 오브젝트의 리스트와 계층 구조를 보여준다.

#### ● 기본적으로는 생성된 순서대로 나열됨

- 드래그해서 순서를 자유롭게 바꿀 수 있음

#### ● 물체 계층 구조 만들기 & 끊기

- 단순히 끌어다 놓는 것으로 가능하다.

#### ● 부모 자식 구조를 만들 수 있음

- 자식은 부모를 따라가지만 부모는 자식을 따라가지 않는다.

### 3) Scene 뷰

#### ● 제작 중인 월드를 편집하는 뷰

#### ● 큐브 만들어 보기

- 메뉴- [GameObject]-[3D Object]-[Cube]



## ● Capsule 이나 Sphere을 생성해보기











## ● Move 툴로 오브젝트를 원하는 위치로 옮겨보기



- 화살표 기즈모를 잡고 이동시켜 보자

## ● 씬 뷰 기즈모 조작하기

- 각 축에 해당하는 원뿔을 클릭하면 그 축으로 화면이 전환된다.
- 기즈모 가운데의 큐브 또는 아래쪽의 Persp를 클릭하면 Perspective, Orthographic 뷰로 전환할 수 있다.

## ● 씬 뷰 조작을 위한 툴 바 인터페이스

-  Hand Tool : 화면을 이동하는 패닝(panning) 동작을 하며 단축키는 Q이다. 어떤 화면 상태든 단축키 없이 마우스 가운데 버튼을 클릭한 채 화면을 움직이면 작동한다.
-  Move Tool : 물체를 움직이는 기능을 하며 단축키는 W이다. 선택된 물체의 3축 기즈모를 클릭하여 물체를 이동할 수 있다.
-  Rotate Tool : 물체를 회전시키는 기능을 하며 단축키는 E이다. 선택된 물체의 3축 기즈모를 클릭하여 물체를 회전시킬 수 있다.
-  Scale Tool : 오브젝트의 스케일을 조절하는 기능을 하며 단축키는 R이다. 선택된 오브젝트의 3축 기즈모를 클릭하여 오브젝트의 스케일을 조절할 수 있다. 기즈모의 가운데 큐브를 이용하면 균등 스케일로 오브젝트를 조절할 수 있다.
-  Rect Tool : 2D 게임이나 UI를 위한 조절자가 나타나며 단축키는 T이다. 2D 모드에서 작업해야 제대로 사용할 수 있다.
-  Move, Rotate, Scale Tool : 앞에서 설명한 Move, Rotate, Scale을 한 번에 조절할 수 있는 툴이 나타나며 단축키는 Y이다.
-  Custom Editor Tool : 커스텀으로 에디터 툴을 지정할 수 있다. 커스텀 에디터가 적용되어 있을 때 물체를 선택한 후 [Scene] 창에서 같은 모양의 버튼을 클릭하면 작동된다.
-  Pivot : Pivot 모드에서는 각 오브젝트의 피봇을 기준으로 스케일되거나 회전한다. Pivot 모드에서 Ctrl을 누른 채 여러 물체를 동시에 선택한 후 회전시키면 다음과 같이 각각의 피봇을 따라 회전한다.
-  Center : Pivot 버튼을 다시 클릭하면 Center 모드로 바뀐다. Center 모드에서 여러 물체를 동시에 선택한 후 회전시키거나 스케일을 조절하면 중심점을 기준으로 회전하거나 스케일된다.
-  Local : Local 모드에서는 각 축의 방향이 물체가 가진 고유의 축에 종속된다.

-  Global: Local 버튼을 다시 클릭하면 Global 모드로 바뀐다. Global 모드에서는 각 축의 방향이 물체와 상관없이 절대 방향을 따른다
-  Snap: 특정 단위 간격으로 물체를 이동할 수 있다.

### ● 확대/축소

- 마우스 휠을 굴린다.
- Alt + 마우스 오른쪽클릭 드래그하면 확대/축소할 수 있다.

### ● 패닝

- 마우스 가운데 버튼을 클릭하여 드래그하면 패닝을 할 수 있다.

### ● 카메라 중심 회전

- 마우스 오른쪽 버튼을 클릭한 채 화면에서 마우스를 움직이면 화면을 회전시킬 수 있다.

### ● 물체 찾기

- Hierarchy창에서 찾고자 하는 물체를 선택하고 더블클릭하거나 F를 누른다.
- Scene창에서 물체를 화면 가득 크게 보고 싶을 때는 물체를 선택하고 F를 누른다.

### ● 물체 중심 회전: 오브젝트를 자세히 관찰할 때 쓰는 기능

- F를 이용하여 특정한 물체를 중심에 놓은 후, 컨트롤 + 마우스 왼쪽 버튼을 클릭하여 드래그

### ● 물체 표면 스냅: 어떤 물체를 다른 물체의 표면 위에 올리하고자 할 때 사용하는 방법

- 다른 물체 위에 올리고 싶은 물체를 선택하고, Ctrl과 Shift를 누르면 나오는 중앙의 사각형 기즈모를 클릭한 채 마우스를 다른 오브젝트 위로 이동시키면 그 표면 위로 물체가 올라가는 것을 볼 수 있다.

### ● 버텍스 스냅: 물체끼리 붙일 때 쓰는 기능(ex: 모듈화된 도로 블록을 붙이기)

- 물체를 선택한 후 V를 누르고 있으면 버텍스를 선택할 수 있으며, 그대로 다른 물체의 버텍스로 옮기면 버텍스끼리 붙는다. 그러나 실제로 물체가 붙는 것은 아니라 버텍스가 동일한 위치로 이동한 것일 뿐이다.

### ● 화면 내비게이션: 게임에서 쓰이는 방식의 화면 내비게이션

- 마우스 오른쪽 버튼을 클릭한 채 키보드의 wasd를 누르면 FPS 게임을 하듯이 화면을 내비게이션할 수 있으며, 이때 Shift를 동시에 누르면 일시적으로 속도가 올라간다. 이는 만들어진 레벨을 탐색할 때 유용한 방법이다.

## ● 씬 뷰 컨트롤 바

- Shaded: 텍스처와 질감이 보인다.
- Wireframe: 와이어프레임 표현으로 메시를 그린다.
- Shaded Wireframe: 텍스처와 질감을 입히고 와이어프레임을 겹쳐 그려준다.
- Shadow Cascades: 디렉셔널 라이트의 색도 캐스케이드를 표시한다.
- Render Paths: 컬러 코드를 사용하여 각 게임 오브젝트의 렌더링 패스를 표시한다. 파란색은 디퍼드 셰이딩, 초록 색은 디퍼드 라이팅, 노란색은 포워드 렌더링, 빨간색은 버텍스 릿을 나타낸다.
- Alpha Channel: 알파로 컬러를 렌더링한다.
- Overdraw: 게임 오브젝트를 투명한 실루엣으로 렌더링한다. 색이 중첩되면 그 부분의 오버드로 수치가 높다는 것을 의미한다.
- Mipmaps: 컬러 코드를 사용하여 이상적인 텍스처 크기를 표시한다. 빨간색은 (현재 거리와 해상도에서) 텍스처가 필요 이상으로 크다는 것을, 파란색은 텍스처가 더 커도 된다는 것을 나타낸다. 이상적인 텍스처 크기는 애플리케이션이 실행되는 해상도와 카메라가 특정 표면에 가까이 근접할 수 있는 정도에 따라 다르다.
- Texture Streaming: 텍스처 스트리밍 시스템의 상태에 따라 게임 오브젝트를 초록색, 빨간색, 파란색으로 채색한다.
- Deferred: G버퍼Geometry buffer의 각 요소(Albedo, Specular, Smoothness, Normal)를 따로 볼 수 있다.
- Global Illumination: UV Charts, Systems, Albedo, Emissive, Irradiance, Directionality, Baked, Clustering, Lit Clustering 등 글로벌 일루미네이션 시스템의 상황을 알 수 있는 많은 정보를 제공한다.
- Material Validation: 물리 기반 매тери얼이 권장 범위 내의 값을 사용하는지 검사할 수 있으며 Albedo와 Metal Specular 모드가 있다.
- 2D: 씬을 2D/3D 뷰로 전환한다. 2D 모드에서 카메라는 +Z축을 향하고 X축은 오른쪽을, Y축은 위를 가리킨다.
- Lighting: 씬의 조명(광원, 오브젝트 셰이딩 등)을 켜거나 끈다.
- Audio: 씬의 오디오 효과를 켜거나 끈다.
- Skybox: 씬의 배경에 렌더링되는 스카이박스를 토글한다.
- Fog: 화면의 안개 효과를 토글한다.
- Flares: 광원의 렌즈 플레어를 토글한다.
- Animated Materials: 셰이더가 애니메이션될 때, 애니메이션되는 매тери얼을 표현해준다.
- Post Processings: 후처리 효과를 토글한다.
- Particle Systems: 파티클 효과를 토글한다.
- 씬 가시성: [Hierarchy] 창에서 숨겨진 오브젝트의 개수를 보여주고 이 기능 전체를 켜거나 끌 수 있다. 씬이 복잡할 때 단순하게 보기 위해 사용하는 이 기능은 Scene 창에만 효과가 있으며 실제 게임에는 영향을 미치지 않는다.
- Field of View: 카메라 뷰의 넓이를 바꾼다. Scene 창에만 효과가 있으며 게임에는 영향을 주지 않는다.
- Dynamic Clipping: 이를 선택하면 유니티가 씬의 뷰포트 크기를 기준으로 카메라의 근

거리 및 원거리 클리핑 평면을 계산한다.

- Clipping Planes: 유니티가 씬의 게임 오브젝트 렌더링을 시작하고 중지하는 카메라와의 거리로 프러스텀 frustum이라고도 한다.

- Near : 유니티가 게임 오브젝트를 렌더링하는, 카메라와 가장 가까운 지점

- Far : 유니티가 게임 오브젝트를 렌더링하는, 카메라와 가장 먼 지점

- Occlusion Culling: 이를 선택하면 Scene 창에서 오클루전 컬링이 활성화되고, 유니티는 다른 게임 오브젝트 뒤에 가려 카메라가 볼 수 없는 게임 오브젝트를 렌더링하지 않는다.

- Camera Easing: 이를 선택하면 카메라는 Duration에 설정된 시간 동안 Scene 창의 모션으로 서서히 움직임을 시작하고 멈춘다. 즉 카메라가 즉시 최고 속도에 이르지 않고 서서히 움직이기 시작하고, 멈출 때도 천천히 멈추게 할 수 있다.

- Duration: 카메라가 최고 속도에 이르는 데 걸리는 시간(단위: 초)으로 Camera Speed에서 설정

- Camera Speed: Scene 창 내 카메라의 현재 속도를 조절한다.

- Min: Scene 창 내 카메라의 최저 속도로 유효한 값은 0.01~98

- Max: Scene 창 내 카메라의 최고 속도로 유효한 값은 0.02~99

- 3D Icons: 컴포넌트 아이콘(Lights, Cameras 등)을 Scene 창에 3D로 그릴지 설정하는 데 사용한다. 이를 선택하면 컴포넌트 아이콘이 카메라와의 거리에 따라 확대/축소되고 씬의 게임 오브젝트에 가려서 보이지 않는다. 아이콘의 전체 크기 표시는 슬라이더로 조정할 수 있다. 이를 선택하지 않으면 컴포넌트 아이콘이 일정한 크기로 표시되고 항상 Scene 창의 게임 오브젝트 위에 표시된다.

- Selection Outline: 선택된 게임 오브젝트가 컬러 아웃라인과 함께 표시되고 자식 게임 오브젝트는 다른 컬러의 아웃라인과 함께 표시된다.

- Selection Wire: 선택된 게임 오브젝트가 와이어프레임 메시로 표시된다.

- Recently Changed: 최근에 수정한 컴포넌트와 스크립트에 대한 아이콘과 기즈모의 가시성을 제어한다.

- Scripts: 씬의 스크립트에 대한 아이콘과 기즈모의 가시성을 제어한다.

- Built-in Components: 아이콘이나 기즈모가 있는 모든 컴포넌트 타입의 아이콘과 기즈모 표시 여부를 설정하는 데 사용한다.

## ● Game 뷰

- 설치되어 있는 카메라를 통해 보이는 장면이 나타나는 창

- 최종으로 빌드된 프로젝트에서 보여주는 화면을 나타낸다.

- 하나 이상의 카메라가 반드시 존재하여야 한다.

- 플레이 모드 : 상단의 플레이 모드 버튼을 클릭하면 지금까지 제작한 게임을 플레이할 수 있다. 플레이 모드 상태일 때 변경한 내용은 일시적이며, 플레이 모드를 종료하면 다시 원래대로 돌아간다. 그러나 메테리얼을 바꾸는 등 Project 창 안의 데이터와 관련된 작업은 돌아오지 않는다.

## ● Game 뷰 컨트롤 바

- Display: 장면에서 여러 카메라가 있는 경우 카메라 목록에서 선택할 때 사용한다.
- Aspect: 다른 모니터에서 게임이 어떻게 보이는지 화면 비율을 테스트하려면 다른 값을 선택한다. Free Aspect가 기본으로 설정되어 있다.
  - Low Resolution Aspect Ratios: 저해상도纵横비를 활성화하여 구형 디스플레이의 픽셀 밀도를 에뮬레이션한다.
  - VSync (Game view only): 수직 동기화 기능을 추가한다. 실제 게임으로 만들었을 때 기능이 추가되는 것이 아니라, 현재 보이는 Game 창에서만 화면 주사율과 그래픽 카드의 갱신 주기를 일치시킨다.
- Scale: 화면을 자세히 보기 위해 확대할 수 있다. Game 창에서 볼 수 있는 이상한 아티팩트를 확인할 때 사용한다.
- Maximize On Play: 플레이 모드를 선택했을 때 화면 전체를 플레이 화면으로 바꿀 수 있다.
- Stats: 오디오와 그래픽에 대한 렌더링 통계를 포함하는 통계 창을 토글할 수 있다.
- Gizmos: 보고 싶은 기즈모를 선택하여 켜거나 끌 수 있다. Scene 창의 기즈모와 같은 동작을 한다.

## ● Inspector 창

- 오브젝트를 하나 선택하면 해당 오브젝트의 컴포넌트가 상세하게 표시된 Inspector 창이 활성화되어 씬에 있는 게임 오브젝트의 기능을 수정, 삭제, 추가할 수 있다.
- Add Component 버튼을 클릭하면 새 컴포넌트를 추가할 수 있다. 컴포넌트에서 마우스 오른쪽 버튼을 클릭하거나 컴포넌트 우상단의 점 세 개를 클릭해서 컴포넌트를 삭제할 수 있다. (컴포넌트의 순서를 바꾸는 것도 가능하다.)

## ● 유니티 프로젝트의 구조

유니티 프로젝트는 씬과 에셋이 존재하며, 씬 안에 게임 오브젝트, 그 안에 컴포넌트로 구성된다.

### 1) Scene

사용자에게 보이는 장면을 의미한다. 씬은 하나일 수도 있고, 여러 개일 수도 있다.  
ex) 로고 씬, 타이틀 씬, 캐릭터 선택 씬, 튜토리얼 등  
새로운 씬을 만들려면 메뉴에서 File-New Scene을 선택한다.

### 2) Game Object

씬을 구성하는 각종 요소를 말한다. 한 씬에는 적어도 하나 이상의 게임 오브젝트가 있으며, 대부분의 경우 많은 게임 오브젝트가 하나의 씬을 이루고 있다. 주로 Hierarchy 창에 있는 것들을 게임 오브젝트라고 한다.

### 3) Component

게임 오브젝트를 구성하고 있는 기능의 집합이다. 오브젝트를 선택하면 Inspector 창에 나온다.

### 4) Asset

씬을 구성하는 데 필요한 모든 리소스를 의미한다. 모든 씬에서 공용으로 사용할 수 있으며, 같은 이름으로는 하나씩만 존재해야 한다. 모델링, 텍스처, 오디오, 애니메이션, 동영상, 텍스트, 폰트, 셰이더, 스크립트 등이 있다. 에셋은 Project 창의 Asset 폴더 안에 위치한다.

### 5) MetaFile

고유 아이디나 해당 파일의 설정 정보를 담고 있는 자동 생성 파일이다. Project 창에서 적당한 에셋을 마우스 오른쪽 클릭하여 Show in Explorer를 선택하면 이를 확인할 수 있다.

#### ● 패키지 : 일종의 zip과 같은 유니티의 데이터 집합 파일

- 에셋 스토어에서 제공하는 에셋 패키지  
에셋 스토어에서 다운받는다. 유니티 최신 버전에서는 패키지 매니저로 통합되었다.
- 플러그인 설치를 위한 유니티 패키지  
유니티 패키지 매니저를 통해 받을 수 있는 공식 패키지이다. 최신 버전에서는 에셋 스토어도 패키지 매니저로 통합되었다.
- 커스텀 패키지  
직접 만들 수 있는 패키지이다. 파일을 다른 위치나 프로젝트로 옮길 때 사용한다. 실무에서는 git이나 svn같은 버전 컨트롤 툴을 사용한다. 유니티에 내장된 Collaborate도 같은 기능을 한다.

#### 1) 에셋 스토어의 에셋 패키지 다운로드

- 브라우저에 에셋 스토어를 검색하고 들어간다.
- 무료 에셋을 다운받는다.
- 내 에셋에 추가하기를 누르고 유니티에 들어가 패키지 매니저를 열면 해당 에셋과 Import 버튼이 생겨있다.
- Import 버튼을 클릭하면 패키지가 다운되고 풀리며 폴더와 파일 구조를 볼 수 있는 창이 뜬다.
- 다시 하단의 Import를 클릭하면 프로젝트에 적용된다.

#### 2) 커스텀 패키지 저장하기

- 커스텀 패키지 저장 : 타 프로젝트로 에셋들을 옮길 때 사용 가능하다.
- Project 창에서 내보내기를 할 오브젝트를 선택한다.
- 마우스 우클릭하여 Export Package를 선택한다.
- Include dependencies : 선택된 오브젝트와 연결된 물체를 전부 포함해서 패키지를 만

- 들 것인지 해당 오브젝트만으로 패키지를 만들 것인지 선택한다.
- Export를 클릭하여 패키지를 저장한다.
- 저장된 패키지 파일을 다른 프로젝트에 드래그해 넣으면 압축이 풀린다.

### 3) 패키지 매니저

- 유니티에 내장된 패키지 관리 시스템이다.
- 각종 플러그인이나 에셋 스토의 패키지를 관리할 수 있는 통합 툴이다.
- Window-Package Manager로 들어갈 수 있다.

## ● Prefab

여러 오브젝트와 컴포넌트가 조합되어 재사용할 수 있도록 만든 독립된 게임 오브젝트를 말한다. 특정한 오브젝트의 개념이 아닌 추상적 개념이다. 각종 에셋들이 모여 하나의 완성된 개념의 오브젝트를 만들 수 있다. 이 프리팹을 프로젝트에 저장하여, 각 씬에서 복제품으로 사용 가능하다.

### 1) 프리팹 만들기

- 캐릭터나 오브젝트에 추가적인 오브젝트등을 넣어서 새로운 작업을 한다.
- 완성된 새 오브젝트를 Project 창으로 드래그하면 새로운 프리팹이 완성된다.

### 2) 프리팹 복제 및 변형

- 인스턴스로 복제될 수 있고 어느 씬에서나 사용 가능하다.
- 프리팹을 더블클릭하거나 Open Prefab을 클릭하면 프리팹 편집 모드가 열린다.
- 프리팹 편집 모드에서 편집 후 나가면 복제 프리팹에 모두 적용된다.
- 복제본 중 하나를 변형시킨 후 Inspector 창에서 Overrides를 선택하면 변화된 내용이 구체적으로 나타난다. 여기서 Revert All을 클릭하면 수정을 취소하고 원본으로 돌아가고, Apply All을 클릭하면 프리팹 원본까지 수정하여 전체 프리팹 인스턴스를 변형시킬 수 있다.

## 5. 게임 시스템 아키텍처 소개

### ● 게임 시스템의 정의

게임의 본질인 게임의 진행이나 재미를 느끼게 하는 총체적인 체계를 말한다.

- 협의의 개념 : 게임의 동작 원리에 해당하는 부분
- 광의의 개념 : 게임의 재미를 주는 모든 요소들

### ● 게임의 장르

게임 시스템의 구조에 따라 분류된 일종의 분류 개념을 말한다.

- 최초 게임 등장기 : 장르가 존재하지 않았다. PC 게임 혹은 비디오 게임이었으며, 게임 종류가 대부분 단순한 맞추기 등의 게임으로 일관되었기 때문에 장르 자체가 필요 없었다.
- 중기 : 게임이 복잡해짐에 따라 다양한 내용으로 새로운 게임을 시도하기 시작했다. 새롭게 시도, 성공한 게임을 바탕으로 하여 비슷한 형태의 아류 작들이 연달아 발표되었다. 비슷한 게임 시스템을 가진 게임들이 등장함에 따라 특정 게임 시스템을 특징으로 하는 게임들이 하나의 군을 이루게 되었다. 특정 게임 시스템을 바탕으로 하는 하나의 군을 장르라는 형태로 발전시키게 되었다.
- 최근 : 장르에 대한 개념이 강화되었다. 그로 인해 장르가 오히려 게임의 시스템을 제약하는 형태로 인식되기도 한다. 그러나 장르는 이미 나온 게임에 대한 분류 기준일 뿐 개발에 적용되는 것은 바람직하지 못하다. 게임의 시스템이 장르에 제한되어서는 안된다.

### ● 게임의 종류

게임의 종류를 여러 기준으로 분류를 할 수 있지만 시스템적으로 분류를 한다면 장르로 분류를 할 수 있다. 장르는 수도 없이 많지만 대표적인 장르만 보자면 다음과 같다.

- 슈팅 게임 : 총을 쏘 적을 격파시키는 게임의 장르다. 한때 PC 게임의 대명사이기도 했다.
- 전략 게임 : 일반적으로 주인공 캐릭터가 등장하지 않고 다수의 유닛이나 장치를 사용하여 전략이나 전술적인 조작을 통해 게임의 목적을 달성하는 게임을 말한다.
- RPG : Level과 경험치가 등장하고, 경험치의 획득을 통해 Level이 상승하며 그에 따라 자신의 캐릭터 역시 강해지고 이를 통해 최종적인 목표를 달성하는 형태의 게임을 말한다.
- 시뮬레이션 : 실제 시스템을 그대로 옮겨와 게임화한 것을 말한다. 모의실험 게임이라고도 말하며, 일반적으로 게임 시스템을 한정하지 않는다. 사실성을 가장 중요한 요소로 본다.



- 어드벤처 : 소설 형태의 게임으로 사용자가 직접적으로 질문과 그에 대한 답을 받아 문제를 추리하고 해결해 나가는 형태의 게임시스템을 말한다. 그래픽의 발전에 따라 문자에서 벗어나 그래픽과 액션적인 요소가 결합한 형태로 발전하였다.
- 스포츠 : 각종 스포츠를 게임으로 옮겨놓은 형태의 게임이다.
- 보드 : 말판이나 패를 가지고 하는 게임을 옮겨 놓은 시스템으로 포커, 주사위, 고스톱 등이 대표적이다.

## ● 게임 디자인의 구성요소-1

게임을 제작하기 위한 설정 및 설계를 말한다.

- 기반 디자인 : 게임의 직접적인 요소는 아니지만, 게임의 전반적인 설정을 위해 기반이 되는 각종 설정 등에 관련된 디자인을 의미한다. 일반적으로 시나리오나 월드 디자인 등이 이에 해당하며 본 디자인을 하기 위한 전 과정에 해당한다.
- 본 디자인 : 게임의 본질에 대한 디자인을 의미한다. 게임의 직접적인 시스템의 디자인이나 레벨 디자인이 이에 해당한다. 실질적인 게임의 디자인이며 핵심이라 할 수 있는 과정이다.
- 실행 디자인 : 디자인된 게임을 실질적으로 구현하기 위한 설정을 의미한다. 게임의 컨셉이나 설정 등에 관해 그래픽이나 사운드 혹은 프로그래밍 구현 가능하도록 설정하는 것을 의미한다.

## ● 게임 디자인의 구성요소-2

위의 3가지 단계를 7개의 세부 과정으로 또 나눌 수 있다.

### - 기반 디자인

월드 디자인 : 게임의 배경이 되는 시대나 배경 혹은 사회 구조, 문화 등을 설정하는 것을 말한다.

시나리오 : 서술적인 형태의 이야기 진행을 말한다. 게임의 줄거리가 필요한 RPG나 어드벤처 장르에서 핵심적인 역할을 하며 다른 장르에서는 주로 게임의 타당성이나 몰입감을 주입하는 요소로 활용된다.

### - 본 디자인

게임 시스템 디자인 : 게임의 동작 과정이나 방법, 진행 형태 등을 디자인하는 것을 말한다. 게임의 핵심이 되는 부분이며 플레이어에게 즐거움을 주기 위한 핵심적인 부분이다.

게임 레벨 디자인 : 게임 시스템과 시나리오 혹은 세계 설정을 바탕으로 실질적으로 게임을 설정하는 것을 말한다. 최근 게임 시스템이 복잡해짐에 따라 매우 중요한 역할이 되었다.

#### - 실행 디자인

게임 그래픽 디자인 : 세부적인 사항은 그래픽 디자이너가 하지만 게임 원안을 창안한 게임 디자이너가 컨셉이나 설정을 제공해 주는 것이 일반적이다. 그와 동시에 인터페이스나 세계관의 이미지 등의 설정도 제공한다.

게임 사운드 디자인 : 과거 큰 요소로 부각되지 않았으나 최근 가장 부각되고 있는 부분 중 하나이다. 전반적인 컨셉에 있어 중요한 부분이다.

게임 프로그램 시스템 디자인 : 게임의 프로그래밍을 위한 디자인을 말하며 일반적으로 프로그래밍에 전문적인 지식이 있는 인력이 한다. 주로 게임 시스템 디자인에 영향을 받으며 게임 그래픽 디자인과 사운드 디자인에 따라 영향을 받기도 하고 제약을 받기도 한다. 게임의 진행이나 설정 및 제약, 구현에 대한 설정을 하는 것이 일반적이다.

### ● 게임 시스템 구성요소

게임 시스템의 핵심은 내부적인 게임의 각종 상호작용 등을 대변하는 게임 파라미터 시스템과 게임의 진행을 결정하는 게임 진행 시스템, 그리고 최종적으로 플레이어의 입력을 받고 출력을 하는 게임 UI 시스템으로 나눌 수 있다. 게임 파라미터 시스템을 바탕으로 게임 오브젝트 시스템과 게임 월드 시스템에 대한 디자인도 필요하다.

#### 1) 게임 시스템

게임의 본질인 게임의 진행이나 재미를 느끼게 하는 총체적인 체계를 말한다. 게임 시스템에 따라 장르를 구분할 수 있으며, 슈팅 게임, RPG, 전략, 시뮬레이션, 스포츠 등 다양한 종류가 존재한다. 게임이 다양해짐에 따라 세분화되고 있다.

#### 2) 게임 디자인

게임을 제작하기 위한 설정 및 설계를 말한다. 게임 기반 디자인을 하는 기반 디자인과 게임의 본질인 시스템과 레벨을 디자인하는 본 디자인, 최종적으로 게임의 구현을 위한 설계를 하는 실행 디자인이 있다. 게임의 동작 과정이나 방법, 진행 형태 등을 디자인하는 것으로 게임의 본질이며 핵심이다.

#### 3) 게임 시스템 구조

파라미터 시스템 : 게임의 이해와 상황 판단을 할 수 있도록 수치적인 시스템으로 구현해 놓은 것을 말한다.

게임 진행 시스템 : 게임의 전반적인 진행에 관련된 순차적인 시스템을 말한다.

게임 월드 시스템 : 게임이 실행되는 환경에 대한 시스템을 말한다.

게임 오브젝트 시스템 : 게임에 등장하는 각종 객체에 관련된 시스템으로 주로 캐릭터 시스템, 캐릭터에 명령을 주는 캐릭터 명령 및 실행 시스템, NPC 시스템, 상호작용 및 성장 시스템, 장비 및 아이템 시스템 등이 그 핵심적인 내용이다.

게임 UI 시스템 : 플레이어가 게임을 하기 위해 입력하거나 그 결과물을 출력받는 시스템을 말한다.

## 6. 모바일 게임 개발

### 1) 기본 에셋 다운로드

- Standard Assets : 기초적인 것들이 대부분 들어있는 에셋이다. 하지만 2017.4 이후 버전부터는 에러가 발생해 에셋 스토어에서 받을 수 없다. 비슷한 다른 에셋을 받아 오거나 강제로 임포트해서 사용할 수 있다.
- 에셋 실행해 보기 : 통합 게임 에셋을 받아올 경우 예제 씬이 포함되어 있다. 해당 씬을 열어 플레이해 볼 수 있다.

처음 씬을 실행 시 에러 메시지들이 보일 수 있으나 자동 수정되어 사라진다.

### 2) Terrain 에디터를 이용한 레벨 디자인

- Terrain 에디터 : 대형 야외 레벨을 제작할 때 사용한다. 작은 방이나 실내 등에는 적합하지 않다.

Game Object - 3D Object - Terrain의 경로로 새 Terrain을 만들 수 있다. Inspector 창에서 여러 설정이 가능하다.

- Fill Heightmap Using Neighbors : 옵션을 끌 경우 주변 Terrain에 영향을 끼치지 않으며 킬 경우 해당 Terrain의 높이에 맞춰 이웃 Terrain의 높이도 어느정도 맞춰진다. 추가로 Mirror 옵션을 킬 경우 해당 Terrain에 맞춰 이웃 Terrain의 형태까지도 어느정도 맞춰진다.

#### - Raise or Lower Terrain

Raise or Lower Terrain을 선택한 후 마음에 드는 브러시를 고르고 크기와 강도를 정하여 드래그하면 Terrain의 높이를 올릴 수 있다. Shift를 누른 채 드래그하면 높이를 낮출 수 있다. (0 이하로는 내려가지 않는다.)

#### - Smooth Height

지형을 부드럽게 다듬고 싶을 때 사용하는 기능이다. 다양한 브러시의 크기와 강도를 선택하여 원하는 지형을 제작할 수 있게 만들어준다.

#### - Stamp Terrain

Raise or Lower Terrain과 마찬가지로 Terrain의 높낮이를 조절할 수 있는 기능이지만, 차이점은 브러시 모양으로 도장을 찍듯이 지형을 찍을 수 있다는 점이다. Ctrl을 누른 채 마우스 휠을 굴리면 높이를 조절할 수 있으며, Subtract 기능으로 빼기도 가능하다. Max <--> Add는 브러시의 높이를 유지하면서 찍을 것인지, 현재 지형에 브러시의 높이를 더할 것인지를 조절하는 옵션이다.

#### - Set Height

지형을 특정 높이로 만드는 기능이다. Terrain은 Heightmap 구조라 0 미만으로 내려가지 않기 때문에, 움푹 파인 계곡이나 강을 만들려면 이 기능을 이용해 처음부터 맵 전체를 일정 높이로 올리고 지형을 제작하기 시작하는 것이 좋다.

#### - Paint Holes

Terrain에 구멍을 뚫는 기능으로, 구멍을 뚫은 주변에 돌 오브젝트를 배치하는 등 경계선 부분을 잘 숨겨서 동굴 같은 것을 만들 때 사용한다. (Shift를 누른 채 드래그하면 뚫린 구멍을 다시 메울 수 있다.)

## ● 지형에 텍스처 입히기

텍스처 레이어 만들기 :

- Paint Texture를 선택한다.
- Edit Terrain Layers를 클릭 후 Create Layer를 선택하여 새로운 텍스처 레이어를 생성한다. (이미 만들어진 레이어가 있다면 Add Layer를 선택하여 가져올 수도 있다.)
- 가져온 Asset에서 텍스처를 한 가지 선택한다. 선택하면 자동적으로 전체 Terrain에 칠해진다.
- 첫 번째 레이어를 생성하면 Project 창에서 확인할 수 있다. 나중에 다른 Terrain에서 Add Layer를 사용해 불러올 수 있다.

나무 배치하기 :

- Paint Trees - Edit Trees - Add Tree를 선택한다.
- 가져온 Asset에 포함된 Tree 프리팹을 선택해 배치할 수 있다.
- 선택 후 드래그해 나무를 배치할 수 있다.
- Shift를 누른 채 드래그해 설치된 나무를 지울 수 있다.
- Tree Height를 조절하여 나무의 크기를 조절할 수 있다.

풀 배치하기 :

- Paint Details - Edit Details - Add Grass Texture를 선택한다.
- 풀이 아니라 작은 돌과 같은 mesh를 배치하고 싶다면 Add Detail Mesh를 선택해서 배치할 수 있다.
- Add Grass Texture 창에서 형식에 맞게 알파 채널이 들어있도록 제작된 텍스처를 선택한다.
- Min/Max Width : 최소 최대 넓이 범위
- Min/Max Height : 최소 최대 높이 범위
- Noise Spread : 풀의 색상을 노이즈 패턴에 맞춰 건강한 풀과 마른 풀 색상으로 약간씩 변화시킬 수 있다.
- Healthy Color : 건강한 풀의 색상을 말한다.
- Dry Color : 마른 풀의 색상을 말한다.
- Billboard : 늘 나를 바라보고 있는 평면의 여부를 말한다.
- 위 설정을 마치고 Add를 눌러 풀을 등록할 수 있다.
- Terrain에 드래그해 설치할 수 있다. Shift를 누른 채 지울 수 있다.
- 이미 설치되어 있는 오브젝트의 설정을 바꾸려면 오브젝트를 선택한 후 Edit Details를 클릭하여 Edit를 선택해서 변경할 수 있다.
- Wind Settings for Grass에서 풀의 흔들림을 제어할 수 있다.

Wind Zone 넣기 :

풀을 제외한 나무는 Wind Zone의 영향을 받는다. 또한 Wind Zone은 파티클에도 영향을 줄 수 있다.

- Game Object - 3D Object - Wind Zone을 선택하여 Wind Zone을 설치할 수 있다.
- Directional : 위치에 상관없이 전체 월드에 영향을 주도록 설치하는 옵션.
- Spherical : 지정한 범위에만 영향을 주도록 설치하는 옵션.

### ● Skybox 메테리얼 직접 만들기

전통적인 스카이 박스 만드는 방법 :

- 6면의 하늘 텍스처를 준비한다.
- 메테리얼을 생성한 후 Skybox/6 Sided 셰이더를 선택하여 각 위치에 맞게 배치한다.
- 사진을 찍거나, 직접 그리거나, 스카이 박스를 렌더링할 수 있는 프로그램을 이용하여 제작한다.

파노라마 이미지를 이용하는 방법 :

- Chrome Ball이나 초광각 렌즈, 360도 특수 카메라 등으로 촬영하고 그래픽 툴에서 이미지를 편집하여 제작하는 방식이다.
- 3D 프로그램에서 렌더링하는 방식도 있다.
- 주로 HDRI가 이용된다. (인터넷에서 Free HDRI를 구할 수 있다.)

### ● Skybox 적용법

- Window - Rendering - Lighting Settings를 선택한다.
- Lighting 창은 자주 쓰이는 창이니 Inspector 창 옆에 가져다 놓는다.
- 다운로드한 에셋이나 만든 메테리얼 중에서 사용할 Skybox 메테리얼을 클릭하여 Lighting 창의 Skybox Material로 드래그한다.
- 검은 음영 부분에 환경광을 적용시키기 위해 Lighting 창의 하단에 있는 Generate Lighting을 클릭한다.

### ● Fog 적용하기

- Lighting 창에서 Other Settings의 Fog를 체크한다.
- 옵션을 켜 후 조금만 멀리 이동하면 Fog가 적용되는지 볼 수 있다. 색상과 밀도 등을 조절해 자연스럽게 만든다.

Fog가 작동하지 않을 때 :

- Scene의 Toggle 메뉴 확인하기
- 렌더링 파이프라인 확인하기 (강제로 Forward로 교체)

### ● 물 설치하기

물을 직접 제작하는 것은 고급 셰이더 기술이 필요하므로 에셋에 내장되어 있는 물 오브젝트를 가져다 배치하는 것이 좋다.

- 에셋의 물 오브젝트를 Scene 창으로 드래그하면 물이 배치된다.
- 물을 배치한 후 Hierarchy 창에서 물을 선택하고 Inspector 창에서 세부 옵션을 조절하며 마음에 드는 물을 만든다.

## ● 조명과 렌즈 플레어

렌즈 플레어는 사진기의 광학적 현상을 흉내 낸 것으로, 카메라에 있는 여러 렌즈의 조합과 조리개, 코팅 등에 의해 생기는 다양한 색과 형태의 부작용이다. 그러나 물리적 렌즈가 없는 게임 엔진에서는 이러한 현상이 일어나지 않으므로 텍스처를 조명 방향으로 나열하는 방법으로 이를 구현한다.

- Hierarchy 창에서 Directional Light를 선택하고 F를 눌러서 Scene 창에 있는 Directional Light를 찾는다.
- 조명을 회전시켜 현재 Skybox에 있는 태양의 방향과 맞게 적당히 맞춘다.(위치는 상관 없으므로 적당한 곳에 놔둔 후 각도만 조절한다.)
- 에셋에서 Flares 파일을 찾고 조명의 Flare에 Flare 파일을 넣는다.
- 빈 오브젝트에 Add Component를 눌러 Lens Flare Component를 적용하고 렌즈 플레어를 등록하면 조명과 상관없는 플레어를 만들 수도 있다.

## 3) 오브젝트 배치와 충돌 체크 설정

### ● 외부 프로그램에서 오브젝트를 가져오는 경우

- 오브젝트는 3Ds Max, Maya, Blender와 같은 외부 DCC 툴에서 가져오는 것이 일반적이다. 고로 3D 그래픽 아티스트가 필요하다.
- 3D 툴에서 FBX 파일로 저장한다.
- 텍스처와 메테리얼은 DCC 툴에서 지정한 것이 넘어오지 않는다. 그래서 엔진으로 텍스처와 FBX 파일을 넘기고 작업한다. 저장한 텍스처와 FBX를 Project 창으로 드래그한다.
- 엔진에서 메테리얼을 생성한다. 알맞은 텍스처를 메테리얼에 지정하여 오브젝트를 완성한다.
- Inspector 창에서 Add Component를 클릭하여 물체의 모양에 적합한 Collider를 추가한다. 물체의 모양이 복잡하고 충돌설정이 섬세해야 할 때는 Collider를 두 개 이상 넣기도 한다.
- 재사용이 가능하게끔 Prefab으로 변환한다.
- 그래픽 작업을 할 사람이 없을 때는 에셋 스토어에서 가져와서 사용할 수도 있다.
- 에셋 스토어에서 가져왔을 경우 설정한 지형과 잘 어우러질 수 있도록 지형을 수정하거나 프리팹의 오브젝트 등을 수정하여 자연스럽게 만들어줘야 한다.
- 나무나 장식, 풀 등으로 경계면을 가려 자연스럽게 만들어주는 것도 좋다.

## 7. 게임의 분류

게임의 수가 많아짐에 따라, 사용자가 게임의 내용을 모두 파악하지 않더라도 손쉽게 게임의 특징을 파악하기 위해 게임의 분류는 아주 중요한 요소가 되었다.

### 1) 게임의 시스템적 특징

게임 장르를 나누는데 가장 핵심이 되는 요소다. 게임의 특징적 시스템에 따른 분류가 있다. ex) RPG, 전략, 슈팅 등

### 2) 시간 시스템

일반적으로 말하는 실시간 시스템과는 다른 게임으로, 게임의 진행을 단절된 개념으로 사용하느냐 혹은 연속된 개념으로 사용하느냐에 따라 구분한다.

- 리얼타임 게임 : 게임 진행이 게임에 관련되는 동작 입력의 끊임과 우선 순위 없이 동시에 진행되는 형식을 말한다.
- 턴 게임 : 입력이 끊기거나 우선 순위가 존재하는 방식의 게임을 말한다. 자신의 차례를 모두 끝낸 후 다음 플레이어나 시스템이 동작하게 되는 것을 말하며 리얼타임 시스템과 반대되는 대표적인 시스템이다.
- 세미 턴 게임 : 리얼타임과 턴 방식의 게임을 적당히 섞어 놓은 게임의 형식을 말한다. 여러 유저가 주고받는 식의 턴 방식이 아니라 각자 자기가 진행을 설정한 후, 모두 자기 턴의 설정이 끝나면 동시에 시간이 실시간으로 흘러 게임의 결과가 나타나고 또 설정하고 실시간으로 진행하는 형태의 세미 턴 방식도 존재한다.

### 3) 소재나 주제

주제에 따른 게임은 주로 소재의 시대적 배경이나 형태로 많이 구분한다. 밀리터리, 무협, 판타지, SF, 액션 등 다양한 소재를 바탕으로 한 게임이 있다.

### 4) 게임 난이도나 깊이

일반적으로 하드코어 게임, 대중 게임이라는 식의 표현을 많이 한다.

### 5) 네트워크 지원

네트워크 지원 여부, 네트워크 통신 방식, 최대 수용 인원에 따라 분류한다.

- Stand Alone 게임 : 네트워크를 지원하지 않는 것
- 네트워크 게임 : 2 ~ 16명 정도가 모여 하는 게임
- 온라인 게임 : 수백 명 이상의 대규모의 인원이 하나의 서버나 Area에 모여 하는 게임

### 6) 플랫폼

게임이 실행되는 환경을 말한다. 예시로 PC용, PS2용, 모바일 등이 있다.

## 8. 게임 플랫폼

게임이 동작하는 기반을 얘기한다. 크게 두가지로 분류된다.

### 1) 하드웨어 플랫폼

게임을 동작시키는 기기를 의미하며, 메인 기기와 입출력 장치, 저장 장치, 네트워크의 유무 등이 속한다. 또한 열린 구조와 닫힌 구조 2가지로 분류할 수 있다.

#### - 열린 구조

열린 구조란 하드웨어 각 부분의 설정이 공개되어 있어 어떤 하드웨어 제작사든 규격에만 맞춰 제작하면 동작하는 형태의 구조를 말한다. 소비자가 원하는 부품으로 하드웨어를 구성할 수 있으며, 필요에 따라 업그레이드가 용이하다. 최신 하드웨어가 가장 먼저 공급되기에 가장 빠른 시간에 사용할 수 있다. 부품만 따로 교체를 할 수 있어 최신 부품이 나오면 해당 부품만 교체도 가능하다.

하드웨어마다 공통적인 규격만 맞춰 다르게 제작되기에 부품간 최적화는 기대하기 힘들다. 특정 부품만 성능이 높고 다른 부품의 성능이 따라가지 못한다면 성능이 낮은 부품의 성능으로 맞춰지는 단점이 존재한다. 부품의 세대가 구세대부터 최신 세대까지 성능과 기능이 다양하기에 개발자의 입장으로 봤을 때 고려할 점이 많다.

#### - 닫힌 구조

닫힌 구조는 열린 구조와 반대로 제품의 모든 구성물이 정확히 맞춰 제작되며 부품을 따로 업그레이드나 변경할 수 없는 제한이 있다. 하드웨어의 공급이 부분적으로 이뤄질 수 없기에 차세대 기기가 나오면 최신 하드웨어를 적용하여 발매되므로 일정 기간 하드웨어는 일정한 성능을 유지하게 된다. 새로 개발되는 하드웨어는 기존 하드웨어와 호환되지 않는 경우가 대부분이다. 새 하드웨어 개발 간격이 짧다면 언제 특정 하드웨어가 사장될지 모르기 때문에 엄청난 비용과 시간이 드는 소프트웨어 개발이 망설여지기 때문에 발매 기간을 짧게 할 수 없다.

각 부품이 처음 하드웨어가 설계될 때 정해지기 때문에 부품 간 최적화가 잘 설정되어 있다. 성능이 최적화할 수 있도록 설정되어 있어 표면적 성능보다 훨씬 좋은 성능을 낼 수 있다. 소프트웨어 개발자 입장에 하드웨어 성능이 일정하기에 성능을 최대한 발휘할 수 있도록 프로그래밍할 수 있다.

### 2) 하드웨어 플랫폼의 고려사항

게임 설계 시 하드웨어의 성능을 반드시 고려해야 한다. 게임의 하드웨어는 게임 시스템을 제한하는 가장 중요한 요소가 될 것이기 때문이다. 개발자는 제작하고자 하는 하드웨어의 제약을 정확히 이해하는 것이 중요하다.

#### ● 주 플랫폼

- CPU의 처리 속도
- 최대 사용 가능 메모리



- 저장 매체 및 용량
- BUS의 성능

#### ● 네트워크 플랫폼

- 프로토콜
- 대역폭, 전송 지연 시간

#### ● 입출력 기기

- 입력 기기의 특성

#### ● 사운드 플랫폼

- 사운드의 채널 수
- 사운드의 처리 능력
- 사운드 메모리

#### ● 비디오 플랫폼

- GPU의 처리 속도
- 비디오 메모리
- GPU 지원 기능

### 3) 소프트웨어 플랫폼

소프트웨어 플랫폼은 일반적으로 운영체제부터 필요한 드라이버나 API 등을 말한다.

## 9. Unity 조명 시스템

### 1) 유니티 실시간 조명 시스템

- 빛은 다양한 현상에 의해 복잡하게 반응한다. 컴퓨터 그래픽에서는 이러한 빛의 성질을 반영하기 위해 ray tracing과 같은 복잡한 계산 공식을 사용하여 긴 시간동안 작업해야 한다. 실시간으로 움직여야 하는 게임 엔진에서는 그렇게 많은 연산을 터릴 수 없다.

- 게임 엔진에서는 자연의 빛 효과를 근사치로 표현하기 위해 여러 개로 나뉜 축약된 기능을 제공한다. 실시간 조명 시스템, 라이트맵, 라이트/리플렉션 프로브 등이 있다.

#### - Directional Light

Directional Light는 멀리서 오는 큰 방향성 조명을 표현하는 데 사용한다. 대표적으로 태양광이 있으며, 특성상 위치는 상관이 없다. 따라서 시작점과 끝점이 없고, 거리가 아무리 멀어도 같은 밝기를 유지한다. Directional Light는 조명의 방향, 강도, 색상만을 제어할 수 있으며 Intensity와 Color는 Inspector 창에서 조절한다.

#### - Spot Light

Spot Light는 손전등, 전등갓, 자동차 헤드라이트 등 현실에서 방향이 제어된 조명을 표현하는데 주로 쓰인다. Spot Light는 Inspector 창에서 Intensity, Color, Range, Spot Angle을 조절할 수 있다.

#### - Point Light

점광원이라고도 불리는 Point Light는 일정 영역을 밝히는 데 유용한 원형 모양의 광원이다. Point Light는 Inspector 창에서 Intensity, Color, Range를 조절할 수 있다.

#### - Area Light

Area Light는 사진관 등에서 흔히 볼 수 있는 넓은 면 형태의 조명을 시뮬레이트하는 데 사용된다. 현실에 흔히 존재하는 이런 부류의 조명은 선명하지 않고 부드럽게 확산되어 실시간으로 계산하려면 너무 많은 연산이 필요하기 때문에 게임 엔진에서는 실시간으로 처리하지 않는다. Inspector 창에서 Color, Range, Area Light의 Width와 Height를 조절할 수 있다.

#### - Ambient Light

야외에서 햇빛을 받지 않는 그늘은 푸른빛을 띠는데 이는 Rayleigh scattering으로 인해 대기에 산란된 푸른빛이 사방에서 들어오기 때문이다. 즉 빛이 닿는 영역만 밝은 것이 아니라 빛이 직접 닿지 않는 영역도 반사와 산란 등에 의해 밝다. 이또한 너무 많은 연산이 필요하기에 게임엔진에서는 실시간으로 처리하지 않는다. 이러한 간접광을 Ambient Light 또는 Global Illumination, GI라고 한다.

Window - Rendering - Lighting Settings에 들어가면 Environment 부분에 Ambient Light에 대한 메뉴가 있다. Lighting 창 하단에 Generate Lighting을 클릭하면 잠시 뒤

환경광이 구워지고 모든 오브젝트에 Skybox 색상의 Ambient Light가 적용된다.

## 2) 유니티 실시간 그림자 시스템

### - CSM (Cascade Shadow Mapping)

대부분의 게임 엔진에서 쓰이는 방식이다. 그림자 자체가 Shadow map이라 불리는 텍스처 연산이므로 아주 넓은 면적을 그리면 해상도가 급격히 떨어지는 단점이 있다.

### - Cascade

가까운 거리와 먼 거리를 나누어 여러 장의 Shadow map을 만드는 것을 말한다. 이렇게 하면 가까운 곳은 선명하게 보이고, 먼 곳은 해상도가 다소 떨어져도 잘 안보이기 때문에 비교적 괜찮다.

### - Soft Shadow

Soft Shadow는 그림자의 외곽 부분을 부드럽게 하는 기능으로, 부하가 커지기는 하지만 그림자의 퀄리티가 높아진다.

## 3) 렌더링 패스

게임 엔진에서 데이터가 입력되어 모니터로 출력되는 과정을 Rendering Path 또는 Rendering Pipeline이라고 한다.

### - Pixel Light

Light 연산이 픽셀 별로 계산되는 가장 섬세하고 복잡한 Light를 말한다. 그림자와 노멀맵 연산을 위해서는 필수적이다.

### - Vertex Light

Pixel Light보다 연산이 간단하고 품질이 낮다. 그림자 연산이나 노멀맵과 같은 고급 기법의 사용 불가능하다. Light에서 Render Mode를 Not Important로 설정하면 강제로 변환 가능하다.

### - Spherical Harmonics Light

복잡한 수학으로 연산되지만 결과물은 약간의 CPU 연산만 필요할 정도로 매우 빠르고 대략적이다. 선명하지 않고 정확하지도 않은, 방향성의 난반사 조명과 같은 결과물이 나온다. 필요 시 엔진에서 자동으로 변환되며, 특수한 경우에 사용된다.

#### 4) 라이트맵

고급 GI 효과는 막대한 연산이 필요하기 때문에 아직은 실시간 처리가 쉽지 않다. 라이트맵은 빛의 복잡한 GI 연산을 텍스처로 구워 오브젝트에 입히는 방식이다. 실시간 조명의 제한 때문에 조명을 사용하지 못하는 환경에서 조명을 표현하여 퍼포먼스를 향상하는 용도로도 사용 가능하다. 단점으로 텍스처 사용량의 증대로 메모리가 증가하고, 미리 Lighting을 구워 놓은 것이라 움직이는 오브젝트에는 적용될 수 없다.

## 10. 게임 장면 시스템

초기 중국 영화에는 시나리오에 Scene은 있었으나 장면 설정은 대화 설정만이 존재했다. 그래서 대화가 아닌 동작 등은 즉석에서 감독이나 연기자들에 의해서 만들어지고, 장면이 줄었다 늘었다 했으며, 대사도 즉석에서 만들어졌다. 하지만 게임에서는 그럴 수 없다. 게임 시스템이 복잡해짐에 따라 게임 제작의 내용이나 과정을 세심하게 설정할 수 있게 해주는 내용들이 필요하다. 이는 스토리나 캐릭터 설정 등의 부분적인 내용이 아니라 게임의 전반적인 구조와 진행에 대한 설정을 말한다. 이러한 것들이 있어야 제한된 비용과 시간 안에서 대규모 게임을 개발할 수 있다.

### ● 게임과 영화의 시스템적인 차이점

- 게임은 상호작용적이다. 영화와 달리 플레이어와 상호작용적으로 진행되고 플레이어의 조작에 의한 반응이 필요하다.
- 게임 진행은 시간의 흐름에 따르지 않는다.
- 게임의 진행은 플레이어의 판단과 조작에 따라 다른 식으로 진행이 될 수 있다. 스토리 전개 같은 부분이 아니라 화면 전환이나 각 장면들의 전환이 플레이어마다 다르다.
- 게임의 주요 목적은 스토리를 플레이어에게 전달하는 것이 아니라 게임 조작에 대한 반응을 전달해 주는 것이 주된 목적이다.

### ● 장면의 표현

#### - 화면 위주의 설정 표현

상태 변화가 직선적이거나 단순하며 주로 시간이나 화면 위주로 진행되는 장면 설정에 많이 사용된다. 콘티 형태나 스토리 보드 형식의 장면 설정이 대표적이다.

#### - 진행 위주의 설정 표현

사용자의 입력을 받거나 선택하는 장면에 가장 많이 사용되는 장면 설정의 형태다. 콘티 형태를 상태 전이도 형태로 구현하는 형태나 완전한 상태 전이도 형태로 나타내기도 한다. 가장 일반적이고 많이 사용되는 설정 표현이다.

#### - 조작 위주의 설정 표현

자주 사용되지 않는 형태다. 조작에 큰 상태 변화가 일어날 정도가 아닌, 아주 단순한 처리의 조작이 나열되어 있거나 그러한 조작이 대부분인 경우 많이 사용된다.

- 분리 설정 표현

3가지 조작을 따로 분리하여 설정하는 것을 말한다. 실제 작업을 위한 이해에 용이하지 않은 표현 방식이나 표현할 내용이 너무 방대하여 한눈에 파악이 힘들 경우 핵심적인 부분만을 추려서 표현하고자 할 때 많이 사용된다.

● 명령 진행 시스템

게임의 진행을 위해 사용되는 각종 신호나 명령에 따른 처리와 진행에 따른 정의를 말한다. 시간 진행 시스템이 거시적인 월드에 적용되는 진행을 정의하는 시스템이라면 명령 진행 시스템은 각각의 개별 객체가 가지는 진행 시스템이다.

● 스토리 진행 시스템

- 스테이지 방식 진행 시스템

각 스테이지를 완수함으로써 게임을 진행해나가는 진행 시스템.

- 캠페인 방식 진행 시스템

스테이지 방식의 변형이자 발전형이다. 단절된 스테이지를 가지고 있으나 하나의 큰 최종 목적을 가지고 그 과정을 단계적이고 비순차적으로 설정해 놓은 방식을 말한다.

- 대화식 진행 시스템

게임 진행을 플레이어의 선택에 따라 직접 혹은 간접적으로 선택하는 형식의 스토리 진행 시스템을 말한다. 어드벤처 게임에서 주로 사용하는 게임 시스템으로 많은 대화와 선택을 통해 스토리를 진행해 나간다.

- 이벤트 방식 진행 시스템

이야기의 전개나 게임의 전개에서 발생하는 각종 사건의 발생을 통한 진행 시스템이다. 어떤 조건이 충족되었을 때 게임의 흥미로운 진행을 위해 사건을 발생시킴으로써 진행의 발전이나 의외성을 부여해 줄 때 많이 사용되는 시스템이다.

- 퀘스트 방식 진행 시스템

플레이어에게 특정한 목표를 주어 그것을 달성하는 과정 즉, 퀘스트를 겪게 하는 진행 시스템이다.

- 무 스토리 진행 시스템

전반적 흐름이나 진행이 없는 게임을 말한다. 무 스토리 게임은 단순히 대전 상대를 바꾸거나 상태만 바꾼 상황에서 반복적인 진행을 하는 형태의 게임을 말한다.

- 진행 방식의 복합

위의 여러 진행 시스템들이 반드시 상충되는 시스템은 아니므로 상황에 따라 여러 가지 진행 시스템을 복합하거나 변형, 혼합하여 사용할 수 있다. 하나의 게임을 플레이 방식에 따라 다양한 진행 방식을 가질 수 있도록 설계하는 경우도 흔하다.